

Geografische Daten visualisieren

Für die Darstellung geografischer Informationen ist die Einbindung eines speziellen Controls, das die geografischen Daten bereit hält, notwendig. Microsoft MapPoint lässt sich vielseitig programmieren und kann daher für zahlreiche Aufgaben genutzt werden. Umsatzzahlen verschiedener Verkaufsregionen oder Störungen an technischen Anlagen lassen sich beispielsweise schematisch zuordnen. Der Anwender kann die Informationen als Karte speichern und damit weiterarbeiten. Alternativ ist es möglich, die Karte als HTML-Datei auf einem Webserver bereitzustellen. Der Artikel führt in die Programmierung von MapPoint ein.

MapPoint kann auf zwei Arten programmiert werden. Sie können ein COM-Add-In entwickeln, genauso wie für die Produkte der Office-Familie, oder Sie verwenden das *Microsoft MapPoint Control*, das sich direkt in Ihre VB6-Anwendung einbetten lässt. Wählen Sie die zweite Variante, haben Sie alle Freiheiten bei der Gestaltung der Benutzeroberfläche. Sie schreiben ein eigenständiges Programm, und nur in einem bestimmten Bereich Ihrer Anwendung wird die Karte von MapPoint dargestellt.

Entwickeln Sie ein Add-In, wird es direkt in der MapPoint-Anwendung ausgeführt, und der Anwender bekommt eine Erweiterung seiner MapPoint-Applikation. Die Benutzeroberfläche von MapPoint kann durch modale Dialoge ergänzt werden.

Im Folgenden wird die Programmierung von MapPoint in einer VB6-Anwendung gezeigt. Die Aufgabe dieser Anwendung ist es, Störungen im Stromnetz geografisch darzustellen. Die Adressen werden aus einer Datenquelle gelesen. Da die Adressangaben oft ungenau sind, wird dem Anwender ein Dialog präsentiert, in dem er auswählen kann, welche Adresse die richtige ist.

Das MapPoint-Control

Fügen Sie über das Menü *Projekt* und den Menüpunkt *Komponenten* einen Verweis auf das *Microsoft MapPoint Control 9.0* hinzu. Dieses Steuerelement erstellt für Sie eine MapPoint-Karte. Im Hintergrund wird eine Instanz von *MapPoint.exe* gestartet, die die eigentlichen Arbeiten übernimmt und die Karte im Steuerelement darstellt. Wenn Sie über den Menüpunkt *Referenzen* die MapPoint-Typenbibliothek importieren, müssen Sie darauf achten,

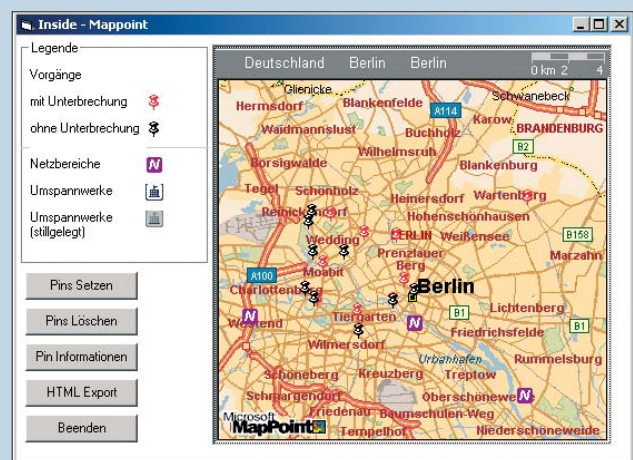


Abbildung 1 | Die Pins markieren Störungen im Stromnetz.

welche Schnittstelle Sie in Ihrem Quellcode verwenden. Mit [F2] erhalten Sie im Objektkatalog beide Typenbibliotheken. Die MapPoint-Typenbibliothek brauchen Sie in der Regel nicht. Einige Funktionen wie beispielsweise der HTML-Export sind im MapPoint-Steuerelement allerdings nicht implementiert.

Daten in MapPoint

Eine Landkarte wird durch ein *Map*-Objekt repräsentiert. Für jedes *Map*-Objekt existiert eine *DataSets*-Auflistung, die alle angezeigten Daten enthält. Es gibt zwei Wege, um Daten auf der Landkarte anzuzeigen. Sie können mit der *ImportData*-Methode Adressdaten direkt aus Excel oder anderen Quellen importieren:

```
Sub OpenDataSet()
    Dim objDataSets As MapPoint.DataSets
    Dim objDataSet As MapPoint.DataSet
    Dim zDataSource As String

    zDataSource = "C:\MeineExcelDatei.xls!Tabelle1!A1:G4"
    Set objDataSets = objApp.ActiveMap.DataSets
    Set objDataSet = objDataSets.ImportData(zDataSource)
End Sub
```

Alternativ dazu lassen sich manuell *PushPin*-Objekte zu einem *DataSet* hinzufügen. In MapPoint werden *PushPin*-Objekte verwendet, um auf der Landkarte an einer bestimmten Position Informationen anzuzeigen. Die *PushPin*-Objekte können verschiedenen Gruppen zugeordnet werden, den *PushPinSet*-Objek-

SUMMARY

Auf einen Blick

MS MapPoint ermöglicht die Visualisierung geografischer Daten. Das MapPoint-Control lässt sich in eigene VB6-Programme einbinden. Der Artikel erläutert an einem Beispiel die Programmierung.

Eingesetzte Anwendungen

VB6 und MapPoint

CD-Code

inside VB02

Autor

Marcel Gnoth ist Senior Consultant bei der Berliner NTeam GmbH. Seine Arbeitsschwerpunkte liegen in den Bereichen COM, .NET und verteilte Informationssysteme, zu denen er auch Trainings durchführt. Sie erreichen ihn unter marcel@gnoth.net oder www.gnoth.net.

ten. Über diese Gruppen können Eigenschaften, wie das Symbol, einheitlich für alle Elemente der Gruppe festgelegt werden:

```
With mpControl.ActiveMap
    'PushPinSets hinzufügen
    Call .DataSets.AddPushpinSet("Set 1")
    Call .DataSets.AddPushpinSet("Set 2")
    'Symbole setzen
    .DataSets.Item("Set 1").Symbol = 1
    .DataSets.Item("Set 2").Symbol = 2
End With
```

Nun können Sie neue *PushPins* erzeugen und den Gruppen zuordnen. In der Beispielanwendung (Abbildung 1) stehen rote Stecknadeln für eine Störung mit einer Unterbrechung des Stromnetzes, schwarze für eine Störung ohne Unterbrechung.

Einen Ort finden

Mit der Methode *FindAddressResults* des *Map*-Objektes wird ein Standort gesucht. Als Parameter können ein Straßename, eine Postleitzahl und der Name der Stadt übergeben werden. Reichen die Angaben nicht aus, um einen Ort eindeutig zu identifizieren, werden mehrere Ergebnisse zurückgegeben. Diese *Location*-Objekte werden in einer *FindResults*-Auflistung gespeichert. Sind mehrere Treffer vorhanden, können diese dem Anwender zur Auswahl angezeigt werden. Über die Eigenschaft *ResultsQuality* der *FindResults*-Auflistung lässt sich feststellen, wie gut die gefundenen *Location*-Objekte zu Ihren Suchkriterien passen. Hat die Eigenschaft den Wert *geoFirstResultGood* wurde ein eindeutiger Ort gefunden und Sie können den Pin setzen (siehe Listing 1). Wenn nicht, können Sie den Anwender entscheiden lassen, welcher Ort der richtige ist.

Listing 1 Adressen suchen.

```
Set m_FindResults = myMap.FindAddressResults(StrassenName, "Berlin", , _
, myPLZ, geoCountryGermany)
If m_FindResults.ResultsQuality = geoFirstResultGood Then
    ' hat nur ein Ergebnis
    Set myloc = m_FindResults.Item(1)
    Set myPin = myMap.AddPushpin(myloc, PinName)

    If Unterbrechung = "N" Then
        myPin.MoveTo .DataSets.Item(PUSH_PIN_SET_OhneUnterbrechung)
    Else
        myPin.MoveTo .DataSets.Item(PUSH_PIN_SET_MitUnterbrechung)
    End If

    myPin.BalloonState = 0
Else
    'Auswahl des richtigen Ortes
End If
```

Nicht eindeutige Adressen auswählen

Dazu wird im Beispielprogramm ein modaler Dialog angezeigt, der in einem Listenfeld die gefundenen Adressen darstellt. Markiert der Anwender eine Adresse mit der Maus, wird auf der Karte ein Fragezeichen-Pin gesetzt. Bestätigt der Anwender die Adresse, wird der Fragezeichen-Pin gelöscht und der endgültige Pin an die Adresse gesetzt. Die *ListView* des Formulars *frmSelectAddress* enthält die Einträge in der gleichen Reihenfolge wie die *FindResults*-Auflistung. Durch einen Klick auf einen Eintrag der *ListView* wird das zugehörige *Location*-Objekt aus der *FindRe-*

sults-Auflistung herausgesucht. Die *GoTo*-Methode des *Location*-Objekts zentriert die Landkarte auf der Adresse, und es wird ein Pin mit einem Fragezeichen-Symbol gesetzt. So erkennt der Anwender auf der Landkarte die nicht eindeutigen Adressen.

Pin-Informationen anzeigen

Die Schaltfläche *Pin Informationen* des Hauptfensters setzt die Beschriftungen für alle Pins der Landkarte. Es gibt drei Varianten: *Keine Beschriftung*, *Nur den Namen anzeigen* oder *Alle Pin-Informationen anzeigen*. Dazu wird die *BalloonState*-Eigenschaft aller Pin-Objekte gesetzt (siehe Listing 2).

Listing 2 Setzen der Pin-Beschriftungen.

```
Private Sub cmdPinShowInfos_Click()
    Dim myRecordset As MapPointCtl.Recordset
    Dim pinDS As MapPointCtl.DataSet
    Dim myPin As MapPointCtl.Pushpin
    Static aktBalloonState As GeoBalloonState

    Me.MousePointer = vbHourglass
    If aktBalloonState = geoDisplayNone Then
        aktBalloonState = geoDisplayName
    ElseIf aktBalloonState = geoDisplayName Then
        aktBalloonState = geoDisplayBalloon
    Else
        aktBalloonState = geoDisplayNone
    End If

    For Each pinDS In mpControl.ActiveMap.DataSets
        If pinDS.Name = PUSH_PIN_SET_MitUnterbrechung Or pinDS.Name = _
        PUSH_PIN_SET_OhneUnterbrechung Then
            'Alle Datensätze für den Datensatz dieses Pins abrufen
            Set myRecordset = pinDS.QueryAllRecords
            Do Until myRecordset.EOF
                Set myPin = myRecordset.Pushpin
                myPin.BalloonState = aktBalloonState
                myRecordset.MoveNext
            Loop
        End If
    Next pinDS
    Me.MousePointer = vbDefault
End Sub
```

Landkarte als HTML-Datei exportieren

Das MapPoint-Steuerelement erlaubt kein Abspeichern einer Karte im HTML-Format. Mit einem Umweg über die MapPoint-Anwendung kommt man dennoch ans Ziel. Das MapPoint-Steuerelement speichert die Karte als MapPoint-Standarddatei (*ptm*). Anschließend wird, für den Anwender unsichtbar, MapPoint als eigenständige Applikation gestartet. Sie lädt die zuvor vom Steuerelement gespeicherte Karte und sichert sie als HTML-Seite.

```
MPApp.ActiveMap.SaveAs FileName, geoFormatHTMLMap, True
```

Die als Erstes gespeicherte PTM-Datei lässt sich nicht löschen, da das MapPoint-Steuerelement diese Datei geöffnet hält. Hierfür müsste ein Workaround geschaffen werden.

Zusammenfassung

MapPoint lässt sich auf vielfältige Art und Weise programmieren. Dennoch stehen nicht alle Funktionen von MapPoint über das Steuerelement zur Verfügung. Im Notfall muss man sich über Umwege behelfen. |||||